# Active Pointillistic Pattern Search

**Yifei Ma**[*] Carnegie Mellon University
**Dougal J. Sutherland**[*] Carnegie Mellon University
**Roman Garnett**[†] Washington University in St. Louis
**Jeff Schneider** Carnegie Mellon University

## Abstract

We introduce the problem of *active pointillistic pattern search* (APPS), which seeks to discover regions of a domain exhibiting desired behavior with limited observations. Unusually, the patterns we consider are defined by large-scale properties of an underlying function that we can only observe at a limited number of points. Given a description of the desired patterns (in the form of a classifier taking functional inputs), we sequentially decide where to query function values to identify as many regions matching the pattern as possible, with high confience. For one broad class of models the expected reward of each unobserved point can be computed analytically. We demonstrate the proposed algorithm on three difficult search problems: locating polluted regions in a lake via mobile sensors, forecasting winning electoral districts with minimal polling, and identifying vortices in a fluid flow simulation.

## 1 Introduction

Consider a function containing interesting patterns that are defined only over a region of space. For example, if you view the direction of wind as a function of geographical location, it defines fronts, vortices, and other weather patterns, but those patterns are defined only in the aggregate. If we can only measure the direction and strength of the wind at point locations, we then need to infer the presence of patterns over broader spatial regions.

Many other real applications also share this feature. For example, an autonomous environmental monitoring vehicle with limited onboard sensors needs to strategically plan routes around an area to detect harmful plume patterns on a global scale [19]. In astronomy, projects like the Sloan Digital Sky Survey [5] search the sky for large-scale objects such as galaxy clusters. Biologists investigating rare species of animals must find the ranges where they are located and their migration patterns [4]. We aim to use active learning to search for such global patterns using as few local measurements as possible.

This bears some resemblance to the artistic technique known as pointillism, where the painter creates small and distinct dots each of a single color, but when viewed as a whole they reveal a scene. Pointillist paintings typically use a denser covering of the canvas, but in our setting, "observing a dot" is expensive. Where should we make these observations in order to uncover interesting regions as quickly as possible?

We propose a probabilistic solution to this problem, known as *active pointillistic pattern search* (APPS). We assume we are given a predefined list of candidate regions and a classifier that estimates the probability that a given region fits the desired pattern. Our goal is then to find as many regions that are highly likely to match the pattern as we can. We accomplish this by sequentially selecting point locations to observe so as to approximately maximize expected reward.

### 1.1 Related Work

Our concept of active pattern search falls under the broad category of *active learning* [15], where we seek to sequentially build a training set to achieve some goal as fast as possible. Our focus solely on finding positive ("interesting") regions, rather than attempting to learn to discriminate accurately between positives and negatives, is similar to the problem previously described as *active search* [6]. In previous work on active search, however, it has been assumed that the labels of interest can be revealed directly. In active pattern search, on the other hand, the labels are never revealed but must be inferred via a provided classifier. This indirection increases the difficulty of the search task considerably.

In *Bayesian optimization* [3, 12], we seek to find the global optimum of an expensive black-box function. Bayesian optimization provides a model-based approach where a Gaussian process (GP) prior is placed on the objective function, from which a simpler acquisition function is derived and op-

Appearing in Proceedings of the $18^{th}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38. 

[*]These two authors had equal contribution.
[†]Work done while at the University of Bonn.

timized to drive the selection procedure. In [17], the authors extend this idea to optimizing a latent function from binary observations. Our proposed active pattern search also uses a Gaussian process prior to model the unknown underlying function and derives an acquisition function from it, but differs in that we seek to identify entire *regions* of interest, rather than finding a single optimal value.

Another intimately related problem setup is that of *multi-arm bandits* [2], with more focus on analysis of the cumulative reward over all function evaluations. Originally, the goal was to maximize the expectation of a random function on a discrete set; a variant considers the optimization in continuous domains [8, 11]. However, like Bayesian optimization, multi-arm bandit problems usually do not consider discriminating a regional pattern.

*Level set estimation* [7, 9], rather than finding optima of a function, seeks to select observations so as to best discriminate the portions of a function above and below a given threshold. This goal, though related to ours, aims to directly map a portion of the function on the input space rather than seeking out instances of patterns. LSE algorithms can be used to attempt to find some simple types of patterns (say, areas with high mean), but even then its learning goal underperforms in the mismatched search objective, and it does not attempt more complex models.

APPS can be viewed as a generalization of *active area search* (AAS) [10], which is a considerably simpler version of active search for region-based labels. In AAS, the label of a region is only determined by whether its mean value exceeds some threshold. APPS allows for arbitrary classifiers rather than simple thresholds, and in some cases its expected reward can still be computed analytically. This extends the usefulness of this class of algorithms considerably.

## 2 Problem Formulation

There are three key components of the APPS framework: a function $f$ which maps input covariates to data observations, a predetermined set of regions wherein instances of function patterns are expected, and a classifier that evaluates the salience of the pattern of function values in each region. We define $f\colon \mathbb{R}^m \to \mathbb{R}$ to be the function of interest,[1] which can be observed at any location $x \in \mathbb{R}^m$ to reveal a noisy observation $z$. We assume the observation model $z = f(x) + \varepsilon$, where $\varepsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. We suppose that a set of regions where matching patterns might be found is predefined, and will denote these $\{g_1, \ldots, g_k\}$; $g_i \subset \mathbb{R}^m$. Finally, for each region $g$, we assume a classifier $h_g$ which evaluates $f$ on $g$ and returns the probability that it matches the target pattern, which we call *salience*: $h_g(f) = h(f; g) \in [0, 1]$, where the mathematical interpretation of $h_g$ is similar to a functional of $f$. Classifier forms are typically the same for all regions with different parameters.

Unfortunately, in general, we will have little knowledge about $f$ other than the limited observations made at our selected set of points. Classifiers which take functional inputs (such as our assumed $h_g$) generally do not account for uncertainty in their inputs, which should be inversely related to the number of observed data points. We thus must consider the probability that $h_g(f)$ is high enough, marginalized across the range of functions $f$ that might match our observations. As is common in nonparametric Bayesian modeling, we model $f$ with a Gaussian process (GP) prior; we assume hyperparameters, including prior mean and covariance functions, are set by domain experts. Given a dataset $\mathcal{D} = (X, z)$, we define

$$f \sim \mathcal{GP}(\mu, \kappa); \qquad f \mid \mathcal{D} \sim \mathcal{GP}(\mu_{f|\mathcal{D}}, \kappa_{f|\mathcal{D}}), \quad (1)$$

to be a given GP prior and its posterior conditioned on $\mathcal{D}$, respectively. Thus, since $f$ is a random variable, we can obtain the marginal probability that $g$ is salient,

$$T_g(\mathcal{D}) = \mathbb{E}_f\big[h_g(f) \mid \mathcal{D}\big]. \quad (2)$$

We then define a matching region as one whose marginal probability passes a given threshold $\theta$. Unit reward is assigned to each matching region $g$:

$$r_g(\mathcal{D}) = \mathbb{1}\big\{T_g(\mathcal{D}) > \theta\big\}. \quad (3)$$

We make two assumptions regarding the interactive procedure. The first is that once a region is flagged as potentially matching (i.e., its marginal probability exceeds $\theta$), it will be immediately flagged for further review and no longer considered during the run. The second is that the data resulting from this investigation will not be made immediately available during the course of the algorithm; rather the classifiers $h_g$ will be trained offline. We consider both of these assumptions to be reasonable when the cost of investigation is relatively high and the investigation collects different types of data. For example, if the algorithm is being used to run autonomous sensors and scientists collect separate data to follow up on a matching region, these assumptions allow the autonomous sensors to continue in parallel with the human intervention, and avoid the substantial complexity of incorporating a completely different modality of data into the modeling process. Making different assumptions would lead to interesting extensions to our algorithms that we do not consider here.

Garnett et al. [6] attempt to maximize their reward at the end of a fixed number of queries. Directly optimizing that goal involves an exponential lookahead process. However, this can be approximated by a greedy search like the one we perform. Similarly, one could attempt to maximize the area under the recall curve through the search process. This

[1] For clarity, in this and the next sections we will focus on scalar-valued functions $f$. The extension to vector-valued functions is straightforward; we consider such a case in Section 4.3.

also requires an intractable amount of computation which is often replaced with a greedy search.

We now write down the greedy criterion our algorithm seeks to optimize. Define $\mathcal{D}_t$ to be data collected before time step $t$ and $\mathcal{G}_t = \{g : T_g(\mathcal{D}_\tau) \leq \theta, \forall \tau \leq t\}$ to be the set of remaining search subjects; we aim to greedily maximize the sum of rewards over all the regions in $\mathcal{G}_t$ in expectation,

$$\max_{x_*} \mathbb{E} \sum_{g \in \mathcal{G}_t} \big[ r_g(\mathcal{D}_*) \mid x_*, \mathcal{D}_t \big], \tag{4}$$

where $\mathcal{D}_*$ is the (random) dataset augmented with $x_*$.

This criterion satisfies a desirable property: when the regions are uncoupled and the classifier $h_g$ is probit-linear, the point that maximizes (4) in each region also minimizes the variance of that region (Section 3.2).

# 3 Method

For the aim of maximizing the greedy expected reward of finding matching patterns, (4), a more careful examination of the GP model can yield a straight-forward sampling method. This method, in the following, turns out to be quite useful in APPS problems with rather complex classifiers. Section 3.1 introduces an analytical solution in an important special case.

At each step, given $\mathcal{D}_t = (X, z)$ as the set of any already collected (noisy) observations of $f$ and $x_*$ as any potential input location, we can assume the distribution of possible observations $z_*$ as

$$z_* \mid x_*, \mathcal{D}_t \sim \mathcal{N}\big(\mu_{f|\mathcal{D}_t}(x_*),\ \kappa_{f|\mathcal{D}_t}(x_*, x_*) + \sigma^2\big). \tag{5}$$

Conditioned on an observation value $z_*$, we can update our GP model to include the new observation $(x_*, z_*)$, which further affects the marginal distribution of region classifier outputs and thus the probability this region is matching. With $\mathcal{D}_* = \mathcal{D}_t \cup \big\{(x_*, z_*)\big\}$ as the updated dataset, we use $r_g(\mathcal{D}_*)$ to be the updated reward of region $g$. The utility of this proposed location $x_*$ for region $g$ is thus measured by the *expected* reward function, marginalizing out the unknown observation value $z_*$:

$$u_g(x_*, \mathcal{D}_t) = \mathbb{E}_{z_*}\big[r_g(\mathcal{D}_*) \mid x_*, \mathcal{D}_t\big] \tag{6}$$

$$= \Pr\big\{T_g(\mathcal{D}_*) > \theta \mid x_*, \mathcal{D}_t\big\}. \tag{7}$$

Finally, in active pointillistic pattern search, we select the next observation location $x_*$ by considering its expected reward over the remaining regions:

$$x_* = \arg\max_x u(x, \mathcal{D}_t) = \arg\max_x \sum_{g \in \mathcal{G}_t} u_g(x, \mathcal{D}_t). \tag{8}$$

For the most general definition of the region classifier $h_g$, the basic algorithm is to compute (6) and thus (8) via sampling at two stages:

1. Sample the outer variable $z_*$ in (6) according to (5).
2. For every draw of $z_*$, sample enough of $(f \mid \mathcal{D}_*)$ to compute the marginal reward $T_g(\mathcal{D}_*)$ in (2), in order to obtain one draw for the expectation in (6).

To speed up the process, we can evaluate (8) for a subset of possible $x_*$ values as long as a good action is likely to be contained in the set.

## 3.1 Analytic Computation of Expected Utility for Functional Probit Models

We now turn to a family of classifiers for which we can compute both (2) and (7) analytically, allowing us to efficiently perform exact searches for potentially complex patterns. This family is formed by a probit link function of any affine functional of $f$.

In the following, we will consider a fixed, arbitrary time step $t$ and omit the time index. Suppose we have observed data $\mathcal{D}$, yielding the posterior distribution $p(f \mid \mathcal{D}) = \mathcal{GP}(f; \mu_{f|\mathcal{D}}, \kappa_{f|\mathcal{D}})$. Let $L_g$ be a linear functional, $L_g \colon f \mapsto L_g f \in \mathbb{R}$, associated with region $g$. (For clarity, we will usually not explicitly notate the dependence on $g$ below.) The family of classifiers is:

$$h_g(f) = \Phi(L_g f + b_g), \tag{9}$$

where $\Phi$ is the cumulative distribution function of the standard normal and $b \in \mathbb{R}$ is an offset. We will consider two specific cases of such functionals in our experiments:

- $Lf \colon f \mapsto w^\top f(\Xi)$, where $\Xi$ is a finite set of fixed points $\{\xi_i\}_{i=1}^{|\Xi|}$, and $w \in \mathbb{R}^{|\Xi|}$ is an arbitrary vector. This mapping applies a linear classifier to a weighted average of $f$ when the input set, and thus $g$, is discrete.
- $Lf \colon f \mapsto \frac{c}{|g|} \int_g f(x)\,\mathrm{d}x$, where $|g|$ is the volume of region $g \subset \mathbb{R}^m$. Here $L_g f$ is the mean value of $f$ on $g$, scaled by an arbitrary $c \in \mathbb{R}$.

Each of these functionals is of the form $\int f(x)\,\mathrm{d}\nu(x)$, but our results below apply to any linear $L$.

As Gaussian processes are closed under linear transformations, $Lf + b$ has a normal distribution:

$$Lf + b \sim \mathcal{N}(L\mu_{f|\mathcal{D}} + b,\ L^2\kappa_{f|\mathcal{D}}),$$

where $L^2$ is the bilinear form $L^2\kappa \colon \kappa \mapsto L\big[L\kappa(x, \cdot)\big] = L\big[L\kappa(\cdot, x')\big]$. For the specific cases above, we can explicitly calculate the mean and variance of $Lf + b$:

$$\begin{cases} \mathbb{E}_f[Lf \mid \mathcal{D}] = w^\top \mu_{f|\mathcal{D}}(\Xi) \\ \mathrm{Var}_f[Lf \mid \mathcal{D}] = w^\top \kappa_{f|\mathcal{D}}(\Xi, \Xi)\, w \end{cases}$$

or

$$\begin{cases} \mathbb{E}_f[Lf \mid \mathcal{D}] = \frac{c}{|g|} \int_g \mu_{f|\mathcal{D}}(x)\,\mathrm{d}x \\ \mathrm{Var}_f[Lf \mid \mathcal{D}] = \frac{c^2}{|g|^2} \iint_{g^2} \kappa_{f|\mathcal{D}}(x, x')\,\mathrm{d}x\,\mathrm{d}x'. \end{cases}$$

For certain classes of covariance functions $\kappa$, the above integrals are tractable; they are encountered when estimating integrals via *Bayesian quadrature*, also known as *Bayesian Monte Carlo* [14].

Then we have the marginal probability (2) in closed form:

$$T_g(\mathcal{D}) = \mathbb{E}_f\big[h_g(f) \mid \mathcal{D}\big] = \mathbb{E}_f\big[\Phi(Lf+b) \mid \mathcal{D}\big] = \Phi\left(\frac{L\mu_{f|\mathcal{D}}+b}{\sqrt{1+L^2\kappa_{f|\mathcal{D}}}}\right). \quad (10)$$

Now we turn to the expected utility (7) of a new observation. Consider a potential observation location $x_*$, and again define $\mathcal{D}_* = \mathcal{D} \cup \big\{(x_*, z_*)\big\}$. Then $u_g(x_*, \mathcal{D})$ is:

$$\Pr\left[\Phi\left(\frac{L\mu_{f|\mathcal{D}_*}+b}{\sqrt{1+L^2\kappa_{f|\mathcal{D}_*}}}\right) > \theta \mid x_*, \mathcal{D}\right] = \Pr\left(\frac{L\mu_{f|\mathcal{D}_*}+b}{\sqrt{1+L^2\kappa_{f|\mathcal{D}_*}}} > \Phi^{-1}(\theta) \mid x_*, \mathcal{D}\right), \quad (11)$$

where $\Phi^{-1}$ is the inverse of the normal CDF. Letting $V_{*|\mathcal{D}} = \mathrm{Var}[z_* \mid \mathcal{D}] = \kappa_{f|\mathcal{D}}(x_*, x_*) + \sigma^2$, we have

$$L^2\kappa_{f|\mathcal{D}_*} = L^2\big[\kappa_{f|\mathcal{D}}(x,x') - \kappa_{f|\mathcal{D}}(x,x_*)\, V_{*|\mathcal{D}}^{-1}\, \kappa_{f|\mathcal{D}}(x_*,x')\big] = L^2\kappa_{f|\mathcal{D}} - L\big[\kappa_{f|\mathcal{D}}(\cdot,x_*)\big]\, V_{*|\mathcal{D}}^{-1}\, L\big[\kappa_{f|\mathcal{D}}(x_*,\cdot)\big], \quad (12)$$

which does not depend on $z_*$. Next, consider the distribution of $L\mu_{f|\mathcal{D}_*}$. If we knew the observation value $z_*$, we could compute the updated posterior mean as

$$\mu_{f|\mathcal{D}_*}(x) = \mu_{f|\mathcal{D}}(x) + \kappa_{f|\mathcal{D}}(x,x_*) V_{*|\mathcal{D}}^{-1}\big(z_* - \mu_{f|\mathcal{D}}(x_*)\big).$$

But, thanks to the linearity of $L$ and the known Gaussian distribution on $z_*$, the updated posterior mean of $Lf$ is also normally distributed with

$$L\mu_{f|\mathcal{D}_*} \mid x_*, \mathcal{D} \sim \mathcal{N}\Big(L\mu_{f|\mathcal{D}}, V_{*|\mathcal{D}}^{-1}\, L\big[\kappa_{f|\mathcal{D}}(\cdot,x_*)\big]^2\Big) \quad (13)$$

and so, using (13) in (11), we can finally compute the desired expected reward $u_g(x_*, \mathcal{D})$ in closed form:

$$u_g = \Phi\left(\frac{L\mu_{f|\mathcal{D}} + b - \sqrt{1+L^2\kappa_{f|\mathcal{D}_*}}\,\Phi^{-1}(\theta)}{\sqrt{V_{*|\mathcal{D}}^{-1} L\big[\kappa_{f|\mathcal{D}}(\cdot,x_*)\big]^2}}\right). \quad (14)$$

### 3.2 Analysis for Independent Regions

The analytical solution to (7) by (14) enables us to further study the theory behind the exploration/exploitation tradeoff of APPS in one nontrivial case: when all regions are approximately independent. This assumption allows us to ignore the effect a data point has on regions other than its own. We will answer two questions in this case:

1. which region will APPS explore next, and
2. what location will be queried for that region.

The key to answer these two questions lies in the fact that because (12) provides a link between $\mathrm{Var}[z_* \mid \mathcal{D}]$ and $\mathrm{Var}\big[L_g f + b \mid \mathcal{D}_*\big]$, there is only one degree of freedom in (14) for a single region. We define this free variable as

$$\rho_g(x_*) = \frac{\sqrt{V_{*|\mathcal{D}}^{-1} L_g\left[\kappa_{f|\mathcal{D}}(\cdot,x_*)\right]^2}}{\sqrt{1+L_g^2\kappa_{f|\mathcal{D}}}} = \big|\mathrm{Corr}\big(z_*,\ L_g f + b + \varepsilon_1 \mid x_*, \mathcal{D}\big)\big|, \quad (15)$$

where $\varepsilon_1 \sim \mathcal{N}(0,1)$ is independent noise.

With this variable, the relation in (12) becomes

$$(1+L_g^2\kappa_{f|\mathcal{D}_*}) = \big(1-\rho_g(x_*)^2\big)(1+L_g^2\kappa_{f|\mathcal{D}}). \quad (16)$$

For $\theta > 0.5$, introduce another shorthand notation[2]

$$R_g = \frac{\Phi^{-1}(T_g(\mathcal{D}))}{\Phi^{-1}(\theta)}, \quad (17)$$

which indicates how close the current state is to the reward; we can rewrite (14) as

$$u_g(x_*, \mathcal{D}) = \Phi\left(\Phi^{-1}(\theta)\,\frac{R_g - \sqrt{1-\rho_g(x_*)^2}}{\rho_g(x_*)}\right). \quad (18)$$

At this step, it is possible to take partial derivatives to find the maximizers for (18). However, the analysis can be made easier if one realizes that, assuming $R_g < 1$, maximizing (18) is equivalent to maximizing the slope of the line joining the following two points $\mathcal{P}_g(x), \mathcal{R}_g$ in $\mathbb{R}^2$:

$$\begin{cases} \mathcal{P}_g(x) = \big(-\rho_g(x), \quad \sqrt{1-\rho_g(x)^2}\big) \\ \mathcal{R}_g = \big(0, \quad R_g\big), \end{cases} \quad (19)$$

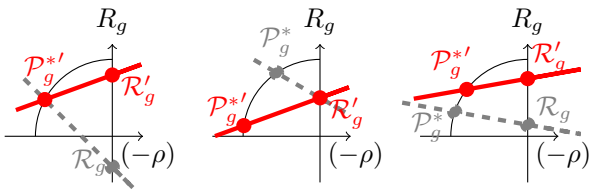
among every available pair of $(g, x)$.

In Figure 1(a), one can observe that the slope of the line can always be made larger by either increasing $\rho_g(x)$, which results in moving the $\mathcal{P}_g(x)$ point to the left along the arc of the unit circle, or by increasing $R_g$.

With the help of Figure 1, we can conclude for regions that do not currently have a reward that

1. For any given region, $u_g(x, \mathcal{D})$ is maximized by choosing the location that yields $\rho_g^* = \max_x \rho_g(x)$.
2. Comparing different regions, if two regions can be equally explored (i.e. they have the same $\rho_g^*$ value), then the region with the larger marginal probability of a matching outcome $R_g$ will be selected. Figure 1(a) illustrates the comparison.

[2] Our conclusions remain the same for any $\theta$; for simplicity, we consider only the common case $\theta > 0.5$ here.




(a) same exploration (b) same mean (c) general criterion

Figure 1: Illustration of selection criterion on independent regions. The solid red line with prime labels is preferred in each plot; it has a larger derivative.

3. If two regions have equal marginal probability of matching the desired pattern $R_g$, then a region with a larger $\rho_g^*$ will be selected. See Figure 1(b).

4. In general, APPS will simultaneously consider both point 2 & 3 (i.e., exploitation and exploration), illustrated by Figure 1(c).

### 3.3 Connection to Active Area Search

We will call the above model with the scaled mean value functional the *mean threshold classifier* (MTC). It is worth pointing out that the MTC is a generalization of the classifier used by Ma et al. [10] for active area search, where a hard threshold was used to decide whether a region is matching. Our sigmoid-based classifier can emulate the step function by taking $|c| \to \infty$ while scaling $b$ appropriately. Closed-form solutions can be found by removing the unit value in the billinearly transformed variance in (10) and (14).

## 4 Empirical Evaluation

We now turn to an empirical evaluation of our framework, in three different settings and with three different classifiers. Code and data for these experiments is available online.[3]

Precision plots are available in the appendix for completeness. Precision is determined primarily by the classifier and $\theta$, and thus does not vary much across methods.

### 4.1 Environmental Monitoring (Linear Classifier)

In order to analyze the performance of APPS with the MTC, we ran it on a real environmental monitoring dataset and compared to baseline algorithms. Valada et al. [19] used small (60 cm) autonomous fan-powered boats to collect dissolved oxygen (DO) readings in a pond, with the goal of finding regions that are low in dissolved oxygen, an indicator of poor water quality. The data used in our experiment comes from a pond approximately 150 meters wide and 50 meters long. The mobile robots have a cell-phone module that records the time and location of every measurement. Because of physical limitations, the measurement reading does not stabilize for about one minute. Therefore, in data collection, the boat was moved back and forth in a single location, in the hope that the noise would cancel by averaging these measurements.

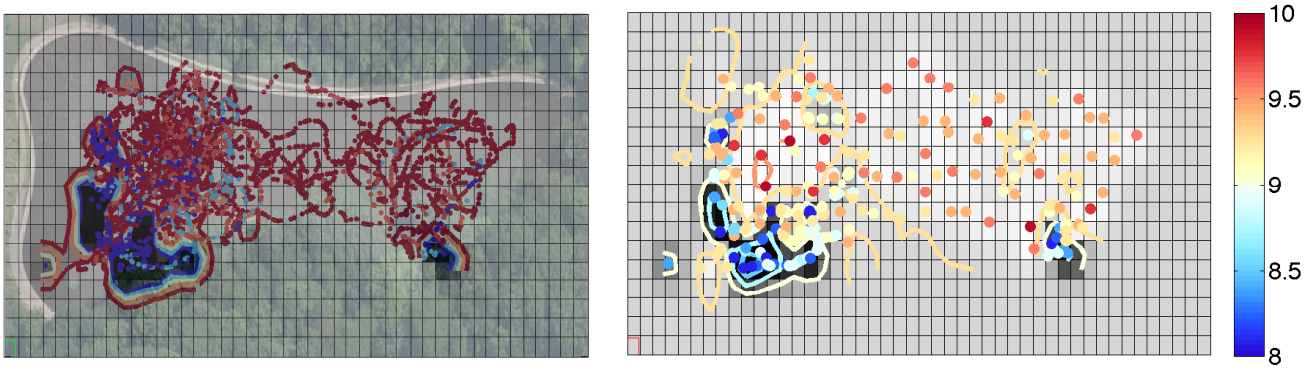
In order to verify our methods, we borrowed data from [19], comprising 16 960 location/DO value pairs, and fit a GP model by maximizing the likelihood of the prior parameters on 500 random samples seven times, taking the median of the learned hyperparameter values. We used a squared-exponential kernel with a learned length scale. We defined regions by covering the map with many windows of size comparable to the GP length scale, and used MTC parameters $b = -9$, $c = -100$. Data points and classifier probability outputs for the ground truth are shown in Figure 2(a), which also shows the learned length scale (roughly 3 meters).

We then repeated the following experiment: we randomly sampled 6 000 points at a time from data points not used for GP parameter training, and randomly selected 10 of these 6 000 points to form an initial training set $\mathcal{D}$. We then used several competing methods to sequentially make further queries until 300 total observations were obtained. The considered algorithms were: APPS with analytical solutions, APPS with one draw of $z_*$ at each candidate location, AAS in Ma et al. [10] with analytical solutions, AAS with sampling, the level set estimation (LSE) algorithm of Gotovos et al. [7] with parameters $\beta^t = 6.25$ and $\varepsilon = 0.1$, uncertainty sampling (UNC), and random selection (RAND). Each algorithm chose queries based on its own criterion; the quality of queried points was evaluated by the MTC classifier with the above parameters and was then compared with true region labels that were computed by MTC using all 6 000 data points. A 70% marginal probability was chosen to be required for a region to be classified as matching ($\theta = 0.7$).

Figure 3 reports the mean and standard error of the recall of matching regions over 15 repetitions of this experiment. APPS and AAS with both analytical solutions and sampling performed equally well here. The similarity between APPS and AAS is also expected because in linear problems, the choice of $c$ is a fine-tuning problem, which does not show its impact on this real dataset. Notice that AAS is not able to handle any other classifier-based setting; this is the core contribution of APPS. To understand why analytical solutions were similar to sampling, notice that the data collection locations have to be constrained to those actually recorded, which makes it easier to obtain a near-optimal decision.

The second group in performance ranking is the LSE method. We attempted to boost its performance by selecting its parameters to directly optimize the area under its recall curve, which was, in a sense, cheating. On further analysis of its query decisions, we saw LSE making, for the most part, qualitatively similar selection decisions to APPS. LSE will stop

[3] https://github.com/AutonlabCMU/ActivePatternSearch/




(a) data in one run and true matching regions (black)

(b) APPS collected data and posterior region probability

Figure 2: Illustration of dataset and APPS selections for one run. A point marks the location of a measurement whose value is also reflected in its color. Every grid box is a region whose possibility of matching is reflected on gray-scale.

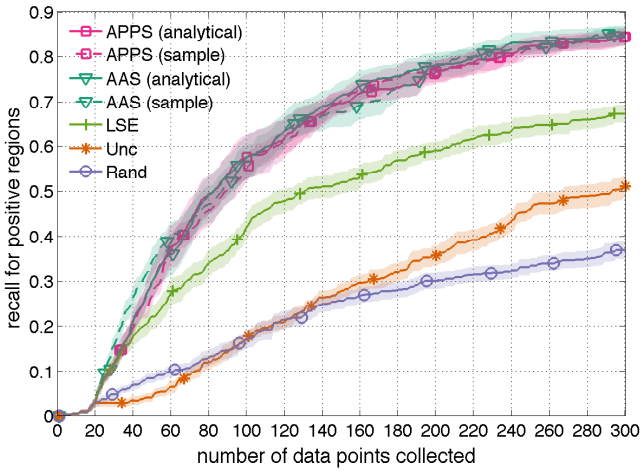



Figure 3: Recall curves for pond monitoring experiment. Color bands show standard errors after 15 runs.

collecting data in a region if there is enough confidence, but does not specifically try to push regions over the threshold, and so its performance on this objective is inferior.

Last in the comparison are RAND and UNC. It is interesting to observe that RAND was initially better than, but later crossed by UNC. In the beginning, since UNC is purely explorative, its reward uniformly remained low across multiple runs, whereas in some runs RAND queries can be lucky enough to concentrate around matching regions. At a later phase, RAND faces the coupon collector's problem and may select redundant boring observations, when UNC keeps making progress at a constant rate.

Figure 2(b) illustrates the selection locations for our APPS method. This plot shows that our APPS method can obtain reasonable data to both explore the available space and gain enough information around the matching regions.

### 4.2 Predicting Election Results (Linear Classifier)

Consider the problem of a state-level political party official who wishes to determine which races will be won, lost, or might go either way. As surveying likely voters is relatively expensive, we would like to do so with as few surveys as possible.

In a simple model of this problem, the problem of finding races which will be won is a natural fit to a classifier of the form $h_g(f) = \Phi\big(w^\mathsf{T} f(\Xi_g) + b_g\big)$. Our function $f$ maps from the voting precincts in the state to the vote share of a given party in that district, with a covariance kernel defined by demographic similarity and geographic proximity. To account for multiple races taking place in each district (e.g., state and national legislators), we duplicate each precinct with a flag for the type of election. If $g$ is the set of all precincts participating in a particular race and $w_g$ is some constant $c$ times the voting population of each precinct, then $w^\mathsf{T} f(\Xi_g)$ gives $c$ times the total vote portion for the given party in that election. In a simple model which ignores turnout effects, the probability of winning a race is essentially 1 if the underlying proportion is greater than 0.5 and 0 otherwise; this can be accomplished by setting $c$ to some fairly large constant, say 100, and $b = -\frac{1}{2}c$. (An equally simple model that nonetheless more thoroughly accounts for unmodeled effects would just use a smaller value of $c$.)

We ran experiments based on this model on 2010 Pennsylvania election returns [1]. For each voting precinct in the dataset, we used the 2010 Decennial Census [18] to obtain a total population count and percentages of the population for gender, race, age, and housing type categories; we also added an $(x, y)$ location based on a Lambert conformal conic projection of point in the precinct, and used these features in a squared-exponential kernel. The data for each precinct was then replicated three times and associated with Democratic vote shares for its U.S. House of Representa-

tives, Pennsylvania House of Representatives, and Pennsylvania State Senate races; the demographic/geographic kernel was multiplied by a positive-definite covariance matrix amongst the races. We learned the hyperparameters for this kernel by maximizing the likelihood of the model on full 2008 election data.

Given the kernel, we set up experiments to predict 2010 races based on surveying an individual voting precinct at a time. For simplicity, we assume that a given voting precinct can be thoroughly surveyed (and ignore turnout effects, voters changing their minds over time, and so on); thus observations were made with the true vote share. We seeded the experiment with a random 10 (out of 16 226) districts observed; APPS selected from a random subset of 100 proposals at each step. We again used $\theta = 0.7$.

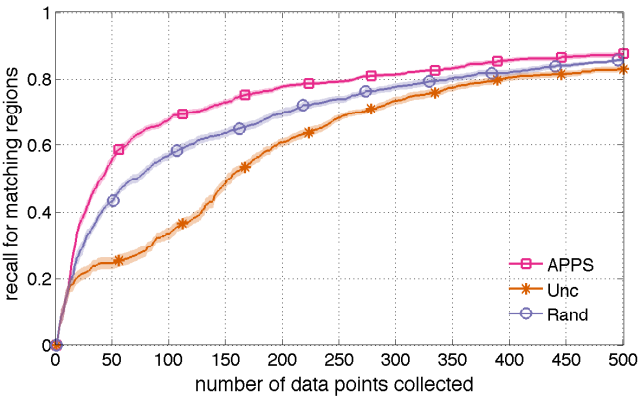



Figure 4: Recalls for election prediction. Color bands show standard errors after 15 runs.

Figure 4 shows the mean and standard errors of 15 runs. APPS outperforms both random and uncertainty sampling here, though in this case the margin over random sampling is much narrower. This is probably because the portion of regions which are positive in this problem is much higher, so more points are informative.

Uncertainty sampling is in fact worse than random here, which is not too surprising because the purely explorative nature of UNC is even worse on the high dimensional input space of this problem.

LSE and AAS are not applicable to this problem, as they have no notion of weighting points (by population).

### 4.3 Finding Vortices (Black-Box Classifier)

We now turn to more complex pattern classifiers by studying the task of identifying vortices in a vector field based on limited observations of flow vectors. Linear classifiers are insufficient for this problem,[4] so we will demonstrate the flexibility of our approach with a black-box classifier.

To illustrate this setting, we consider the results of a large-scale simulation of a turbulent fluid in three dimensions over time in the Johns Hopkins Turbulence Databases[5] [13]. Following Sutherland et al. [16], we aim to recognize vortices in two-dimensional slices of the data at a single timestep, based on the same small training set of 11 vortices and 20 non-vortices, partially shown in Figure 5(a).

Recall that $h_g$ assigns probability estimates to the entire function class $\mathcal{F}$ confined to region $g$. Unlike the previous examples, it is insufficient to consider only a weighted integral of $f$. Instead, though, we can consider the average flow across sectors (angular slices from the center) of our region as building blocks in detecting vortices. We count how many sectors have clockwise/counter-clockwise flows to give a classification result, in three steps:

1. First, we divide a region into $K$ sectors. In each sector, we take the integral of the inner product between the actual flow vectors and a template. The template is an "ideal" vortex, but with larger weights in the center than the periphery. This produces a $K$-dimensional summary statistic $L_g(f)$ for each region.

2. Next, we improve robustness against different flow speeds in the data by scaling $L_g(f)$ to have maximum entry 1, and flip its sign if its mean is negative. Call the result $\tilde{L}_g(f)$.

3. Finally, we feed the normalized $\tilde{L}_g(f)$ vector through a 2-layer neural network of the form

$$h_g(f) = \sigma\left(w_{\text{out}} \sum_{i=1}^{K} \sigma\left(w_{\text{in}} \tilde{L}_g(f)_i + b_{\text{in}}\right) + b_{\text{out}}\right),$$

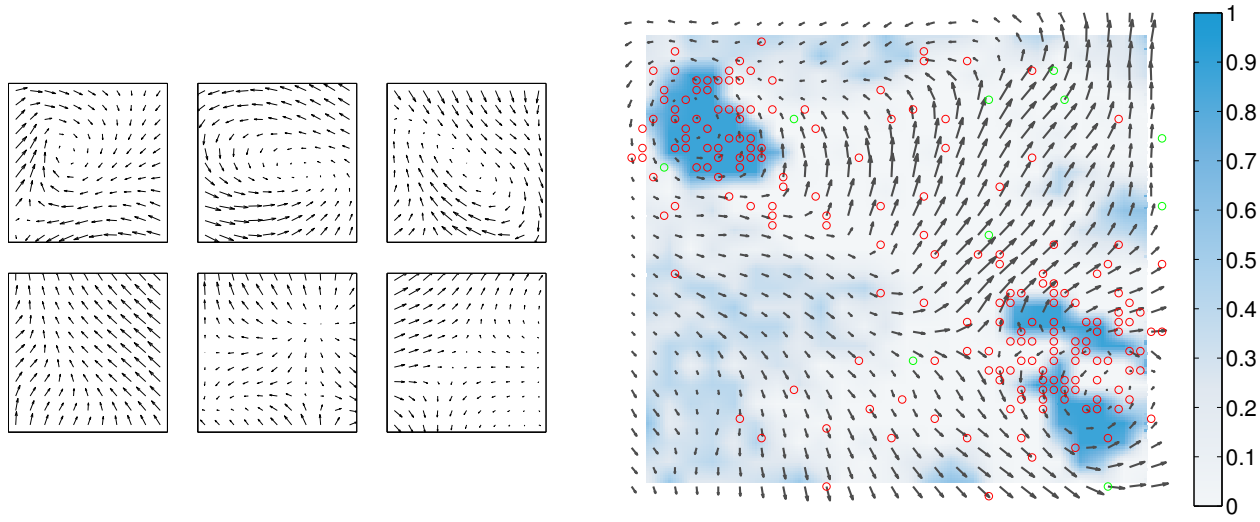
   where $\sigma$ is the logistic sigmoid function.

$L_g(f) \mid \mathcal{D}$ obeys a $K$-dimensional multivariate normal distribution, from which we can sample many possible $L_g(f)$, which we then normalize and pass through the neural network as described above. This gives samples of probabilities $h_g$, whose mean is a Monte Carlo estimate of (2).

We used $K = 4$ sectors, and the weights in the template were fixed such that the length scale matches the distance from the center to an edge. The network was optimized for classification accuracy on the training set. We then identified a $50 \times 50$-pixel slice of the data that contains two vortices, some other "interesting" regions, and some "boring" regions, mostly overlapping with Figure 11 of Sutherland et al. [16]; the region, along with the output of the classifier when given all of the input points, is shown in Figure 5(b). We then ran APPS, initialized with 10 uniformly random points, for 200 steps. We defined the regions to be squares of size $11 \times 11$ and spaced them every 2 points along the grid, for 400 total

[4] The set of vortices is not convex: consider the midpoint between a clockwise vortex and its identical counter-clockwise case.

[5] http://turbulence.pha.jhu.edu




(a) (b)

Figure 5: (a): Positive (top) and negative (bottom) training examples for the vortex classifier. (b): The velocity field used; each arrow is the average of a $2 \times 2$ square of actual data points. Background color shows the probability obtained by each region classifier on the 200 circled points; red circles mark points selected by one run of APPS initialized at the green circles.

regions. We again thresholded at $\theta = 0.7$. We evaluate (2) via a Monte Carlo approximation: first we took 4 samples of $z_*$, and then 15 samples from the posterior of $f$ over the window for each $z_*$. Furthermore, at each step we evaluate a random subset of 80 possible candidates $x_*$.

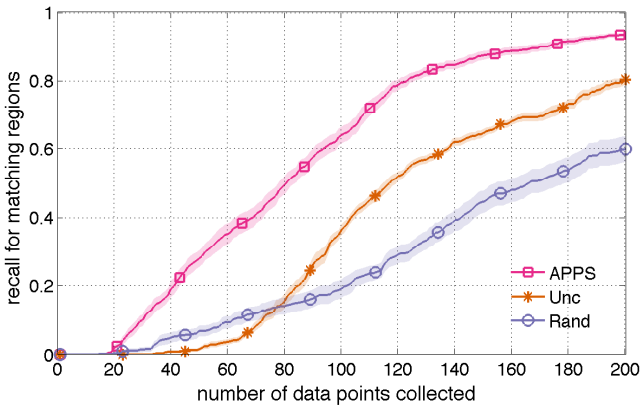



Figure 6: Mean recalls over the search process on the vortex experiment. Color bands show standard errors after 15 runs.

Figure 6 shows recall curves of active pattern search, uncertainty sampling, and random selection, where for the purpose of these curves we call the true label the output of the classifier when all data is known, and the proposed label is true if $T_g > \theta$ at that point of the search (evaluated using more Monte Carlo samples than in the search process, to gain assurance in our evaluation but without increasing the time required for the search). We can see that active pattern search substantially outperforms uncertainty sampling and random selection. As in Section 4.1, uncertainty sampling was initially bad but later surpassed random selection, for the same reason.

## 5 Conclusions

We have introduced the general active pointillistic pattern search problem, where we seek to discover specific local patterns exhibited by an underlying smooth function with a limited observation budget. We proposed a framework built on Bayesian decision theory for the sequential active selection of observations so as to maximize the expected number of matching locations discovered at termination. We derived analytical forms for the required quantities for a broad class of models, and demonstrated the method's efficacy across three very different settings, using two different analytical classifier forms and one based on sampling.

### Acknowledgements

This work was funded in part by DARPA grant FA87501220324 and by the German Science Foundation (DFG) under reference GA 1615/1-1.

## References


[1] S. Ansolabehere and J. Rodden. Pennsylvania data files. URL http://hdl.handle.net/1902.1/16389.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[3] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. arXiv:1012.2599 [cs.LG].

[4] J. L. Brown, A. Cameron, A. D. Yoder, and M. Vences. A necessarily complex model to explain the biogeography of the amphibians and reptiles of Madagascar. *Nature communications*, 5:5046, Jan. 2014. ISSN 2041-1723.

[5] D. J. Eisenstein, D. H. Weinberg, E. Agol, H. Aihara, C. Allende Prieto, and et al. SDSS-III: Massive spectroscopic surveys of the distant universe, the Milky Way, and extra-solar planetary systems. *The Astronomical Journal*, 142:72, Sept. 2011.

[6] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. P. Mann. Bayesian optimal active search and surveying. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, 2012.

[7] A. Gotovos, N. Casati, G. Hitz, and A. Krause. Active learning for level set estimation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2013.

[8] O. B. Kroemer, R. Detry, J. Piater, and J. Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous Systems*, 58(9): 1105–1116, Sept. 2010.

[9] K. H. Low, J. Chen, J. M. Dolan, S. Chien, and D. R. Thompson. Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 105–112, Richland, SC, 2012.

[10] Y. Ma, R. Garnett, and J. Schneider. Active area search via Bayesian quadrature. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, 2014.

[11] S. Niranjan, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 2010.

[12] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In *Proceedings of the 3rd Learning and Intelligent Optimization Conference (LION 3)*, 2009.

[13] E. Perlman, R. Burns, Y. Li, and C. Meneveau. Data exploration of turbulence simulations using a database cluster. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomupting*, 2007.

[14] C. E. Rasmussen and Z. Ghahramani. Bayesian monte carlo. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2003.

[15] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[16] D. J. Sutherland, L. Xiong, B. Póczos, and J. Schneider. Kernels on sample sets via nonparametric divergence estimates, 2012. arXiv:1202.0302 [cs.LG].

[17] M. Tesch, J. Schneider, and H. Choset. Expensive function optimization with stochastic binary outcomes. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, 2013.

[18] United States Census Bureau. 2010 Census, 2010. URL http://www.census.gov/2010census/data/.

[19] A. Valada, C. Tomaszewski, B. Kannan, P. Velagapudi, G. Kantor, and P. Scerri. An intelligent approach to hysteresis compensation while sampling using a fleet of autonomous watercraft. In *Intelligent Robotics and Applications*, volume 7507 of *Lecture Notes in Computer Science*. 2012.